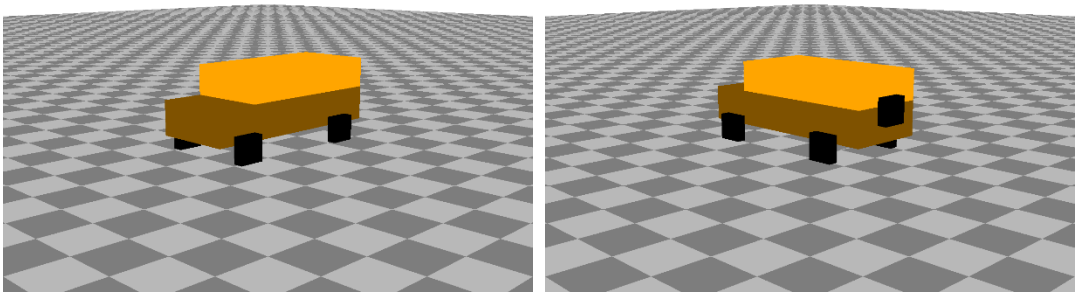
Proseminar Visual Computing
Winter Semester 2021

CG Assignment 1
Hand-out: November 23, 2021
Hand-in: December 7, 2021



Topics

- General OpenGL programming
- Transformations
- Basic animations and user controls
- Camera control

Outline

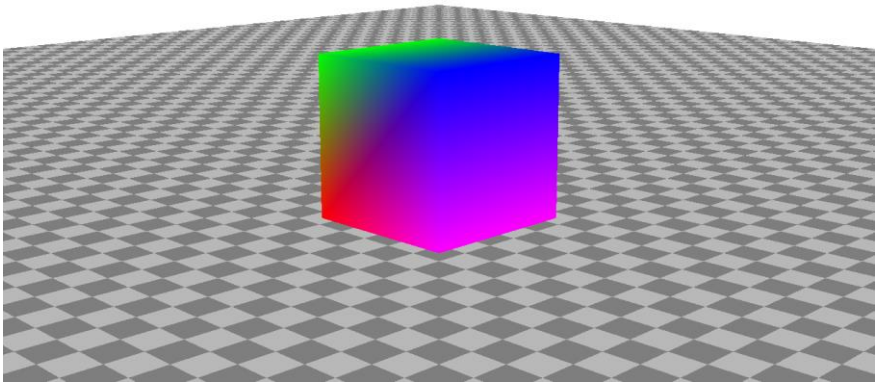
The goal of the Computer Graphics assignments of the Visual Computing PS is to build an animated car. This work is divided in 3 steps. Each step corresponds to a programming assignment. In this first assignment, we focus on geometric transformations and animation. The objective of this first assignment is to build a very simple car made from basic geometric primitives (i.e., cubes) by applying different transformations. Further, the car should be controlled using user input and when driving around, the wheels should be animated. Finally, an additional camera mode should be added that follows the car. See the example video provided with this assignment for a working solution including all features.

Template code

A template code is provided for this assignment. You can modify this code to build your own scene. The functions to modify are all implemented in the main source file **assignment_1.cpp**.

The scene available in the template code contains a base plane and one cube that can be rotated around its x- and its y-axis using the W, S, A, D keys. Moreover, an orbit camera with

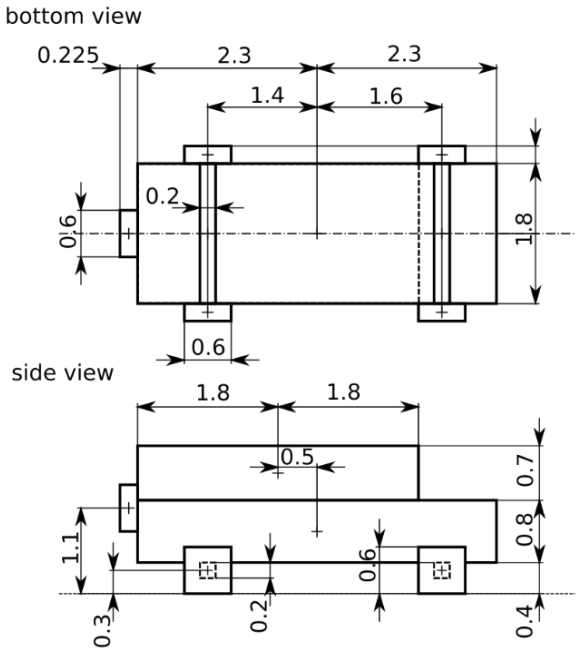
the cube in its center is available. The camera can be controlled using the mouse by pressing the left mouse button. Then it follows the mouse movements and rotates around the cube on a sphere with a fixed radius. To zoom in and out (i.e., de- and increase the radius of the sphere) the mouse wheel is used. In the figure below, an example of the template scene is depicted:



Tasks

1. Setup the hierarchical geometrical model of the car. For that, disable the scaled cube model from the template code. Then start with **9 unit-cubes** (size: 1x1x1) and apply transformations to them (**scaling, rotation, and translation**) to obtain the **car parts** representing: **base, roof, front and back axis, wheels, and spare tire** (see introduction figure). All the necessary measurements for the different parts can be taken from the drawing below. You are free to design your own car, but it must include **at least the same number of parts**. In addition, the transformations **scaling, rotation,** and **translation** must be used for its creation.

   Use **different colors for the different parts** (see example images).



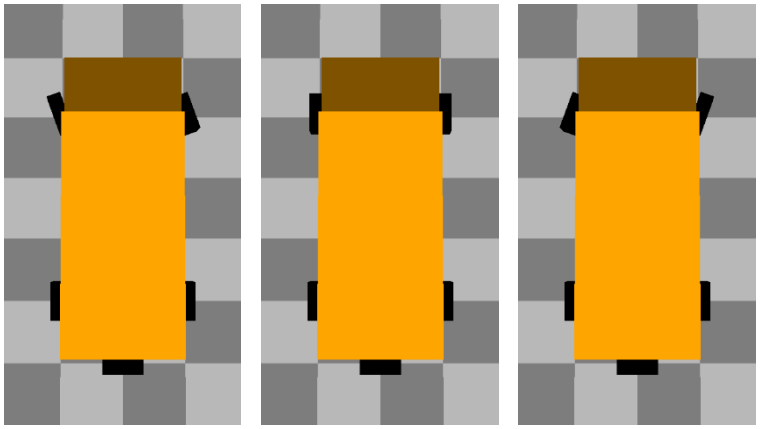
2. Add **user control** and **animation** to the car. It should be possible to drive the car around using the **W, A, S, D keys**. In addition, the **car wheels** should be **animated** and should **turn according to the car's movement**.

   - In a first step, it should be possible to control the car's forward and backward movement by pressing W and S, respectively. The car should move with a predefined velocity (e.g., 8.33 m/s = 30 km/h). Using the velocity together with the elapsed time since the last update step, the distance the car moves can be obtained. Further, to calculate the rotation angle α of the wheels, use the following formular:

     $$\alpha = \frac{\Delta x}{r}$$

     with r the wheel radius and Δx the travelled distance.

   - In a second step, the possibility to control the car steering, as well as the steering animation should be added. When pressing A/D, the front wheels of the car should turn left/right for a predefined turning angle θ (see example figure below). Further, if the car moves forward or backward, it should turn. To calculate the angle that the car turns for every meter it travels, use the method **carCalculateTurningAnglePerMeter(…)**. It takes as input the turning angle of the car (**carTurningAngle**), the distance between the wheels (**carWheelBase**) and the width of the car (**carWidth**). This constant can then be used to calculate how much the car turns in each update step depending on the travelled distance.



3. Finally, implement a second camera mode. The first mode is very similar to the camera available in the template code. But instead of initially looking at the center of the cube, the camera should initially look at the origin of the world coordinate system (i.e., $(0,0,0)^T$). Further, in the second mode the camera focus should follow the car. Use the keys **1** and **2** to switch between the two modes. If switching from mode 2 to mode 1,

it is not required to reset the focus to the origin, but just keep the last focus point from mode 2.

Implementation Remarks

Make sure that your code is clear and readable. Write commentaries when necessary. Your solution should contain a readme file with names of the team members, list of keyboard controls, and any explanation that you think is necessary for the comprehension of the code.

Submission and Grading

Submission of your solution is due on December 7th, 2021 (23:59). **Submit the sources** (i.e., only the content of the *src* folder) in a ZIP archive via OLAT. Do not submit the executable and the content of the *build* folder. Do not submit the external dependencies either. Both folder and archive should be named according to the following convention:

Folder: **CGA1_<lastname1>_<lastname2>_<lastname3>**

Archive: **CGA1_<lastname1>_<lastname2>_<lastname3>.zip**,

where <lastname1>, etc. are the family names of the team members. Development in teams of two or three students is requested. Please respect the academic honor code. In total there are 15 marks achievable in this assignment distributed as follows:

- Geometrical models (**5.5 marks**)
- Car control and animation (**5 marks**)
- Additional camera mode **(2.5 marks)**
- Code readability, comments, and proper submission: (**2 marks**)

Resources

- Lecture and Proseminar slides as well as code and information are available via OLAT.
- OpenGL homepage
  http://www.opengl.org
- OpenGL 3.3 reference pages
  https://www.khronos.org/registry/OpenGL/specs/gl/glspec33.core.pdf
- OpenGL Tutorial
  https://learnopengl.com/
  http://www.opengl-tutorial.org
- GL FrameWork GLFW
  https://www.glfw.org/documentation.html

Note: Be mindful of employed OpenGL and GLSL versions!